Algorithms
Computer Science 140 & Mathematics 168
Spring 2018
Homework 1
Due: Tuesday, January 23 at 11:59 PM

- Please read *Handout 1: Good Writing, Induction, and Greed.* You'll notice that the grading rubrics on Canvas allocate a significant number of points on each problem to the writing objectives outlined in that handout.
- Please submit each problem separately in pdf format on the Canvas system.
- Each problem on this assignment must be typeset using LaTeX. You are welcome to start with the LaTeX source files posted on Canvas and modify them. Remember to submit a pdf of the typeset document, not the `.tex` file. *Please* remember to use the default margins. In other words, don't use a margin package such as `geometry` or `fullpage`.
- At the end of each problem, please include a brief self-assessment. If you know that your solution is not quite complete or correct, let us know that you recognize that and where you believe that the issue arises. If you're confident in your solution, let us know that as well! Generally, just one sentence suffices here. Part of your score will be based on an accurate self-assessment of your work.
- Bonus problems are optional but you are strongly encouraged to try them. They will help you develop a deeper level of mastery of the material in this course.

Problem 1: Properties of Logs. [15 Points]

The objective of this problem is to remind you of some important identities of logarithms that we'll be using next week and throughout the course.

Let's begin by proving that $\log_b xy = \log_b x + \log_b y$. The proof goes as follows: Let $k = \log_b xy$, $\ell = \log_b x$, and $m = \log_b y$. Then, $b^k = xy$, $b^\ell = x$, and $b^m = y$, by the definition of the logarithm. Thus, $b^k = b^\ell b^m = b^{\ell+m}$ by properties of exponents. Thus, $k = \ell + m$, which is what we had set out to prove.

Now give proofs for each of the following properties of logarithms. Write your proofs out carefully. You should assume that a, b, c, n are positive real numbers (*not necessarily integers*).

a. $\log_b a^n = n \log_b a$.

b. $\log_b a = \frac{\log_c a}{\log_c b}$.

c. $a^{\log_b n} = n^{\log_b a}$. (This result is sometimes called the "log-switching theorem" since it says that we can "switch" a and n.)

Solution:

a. *Proof.* Notice that we cannot assume that n is an integer. The problem stipulates that it is a positive real number! So, let $x = \log_b a$. Then $b^x = a$.

$$b^x = a \implies b^{xn} = a^n \implies \log_b b^{xn} = \log_b a^n \implies xn = \log_b a^n \implies n \log_b a = \log_b a^n$$

□

b. *Proof.* Let $x = \log_b a$. Then $b^x = a$.

$$b^x = a \implies \log_c b^x = \log_c a \implies x \log_c b = \log_c a \implies x = \frac{\log_c a}{\log_c b} \implies \log_b a = \frac{\log_c a}{\log_c b}$$

□

c. *Proof.* Let $x = \log_b n$ and let $y = \log_b a$. Then $b^x = n$ and $b^y = a$. So, $a^{\log_b n} = a^x$. Since $a = b^y$, we have $a^{\log_b n} = (b^y)^x = b^{xy}$. Similarly, $n^{\log_b a} = (b^x)^y = b^{xy}$. Therefore, $a^{\log_b n} = n^{\log_b a}$. □

Problem 2: Hurts Car Rental! [25 Points]

Hurts Car Rental is experimenting with a new type of environmentally-friendly vehicle. These vehicles use a modular fuel pack[1] that allows the vehicle to travel exactly R miles for some value R that depends on the vehicle. When the car needs to be refueled, the fuel pack is removed from the car and is replaced by a new one. Thus, it is possible that a fuel pack will be replaced before it is completely used up. Moreover, the driver gets no credit for the remaining fuel in the fuel pack and can only purchase a new fully charged R-mile pack.

Consider a highway represented by a line segment. Let $p_1, p_2, \ldots, p_n$ be n towns (points) on the line segment sorted from left to right where p_1 is the starting point, p_n is the destination point, and each of these towns has a service station selling fuel packs. We are given an array D of length n where $D[1] = 0$ and for $1 < i \leq n$, $D[i]$ is the distance from town p_{i-1} to p_i. You may assume that the car begins at p_1 with a new fuel pack and that the distance between any two adjacent cities is at most R, the cruising range of the vehicle on a fuel pack.

[1]The fuel is believed to consist of a mixture of liquified Spam, Mountain Dew, and Cheetos, although the technology is proprietary and this is just speculation.

a. Describe a greedy algorithm that determines a smallest set of towns where fuel packs can be purchased in order to arrive at the destination. There is no need to purchase a tank at the destination.

b. Prove the correctness of your algorithm using mathematical induction and the safe choice method.

Solution:

a. The algorithm begins at point p_0 and finds the largest i such that the distance from p_0 to p_i is less than or equal to R. It refuels at p_i (adding that city to the initially empty list of refueling cities) and repeats the process beginning at p_i.

b. The proof is by strong induction on the number of towns, n. The base case is when $n = 0$ in which case our algorithm is trivially correct. For the induction hypothesis, assume that the algorithm is correct for 0 to n towns and now, in the induction sets, consider an arbitrary set of $n + 1$ towns. Let p_i be the city that our greedy algorithm selects for the first refueling. We first claim that there exists an optimal solution that includes p_i. Let OPT denote an optimal solution, that is, a set of refueling cities that is valid and for which no smaller valid set exists). If $p_i \in$ OPT then there exists an optimal solution that contains p_i. Otherwise, let p_j be the city with smallest index in OPT. Note $j < i$ since $j \neq i$ by assumption and if $j > i$ then p_j is necessarily more than R away from p_0 and the solution is invalid. Therefore, we can replace p_j with p_i and we still have a valid solution because p_i is reachable from p_j and the next city in OPT is closer to p_i than it is to p_j. Thus, there exists an optimal solution that includes the town p_i selected by the greedy algorithm. That optimal solution must now find the optimal number of towns to refuel to get from p_i to p_n. But that problem has between 0 and n towns and, by the strong induction hypothesis, our greedy algorithm solves that remaining problem optimally.

Problem 3: Scary Objects! [30 Points]

You're on a spaceship in a 2D universe. Periodically, scary objects appear around you. Your objective is to zap them with a special zapper. A zap is simply a ray emanating from your spaceship. That ray is infinitely long and will annihilate all objects with which it intersects. (Intersection at a point is considered a valid intersection.) However, each zap uses a tremendous amount of energy, so your objective is to use the least number of zaps to obliterate the scary objects around you.

Here's what you're given: Your location is assumed to be at the origin of the coordinate system. Each of the n scary objects is represented by two vectors such that any ray fired between these two vectors (including these two vectors themselves) will annihilate the object. Each vector is represented by its angle (between 0 and 360 degrees) counterclockwise from the x-axis (which is the horizontal line through your spaceship). The ordered pair of angles

(θ, ϕ) for an object are specified so that $\theta < \phi$. For example, Figure 1 shows five objects and pairs of vectors representing the angular extent of each of those objects. The blue hexagon in Figure 1 is represented by the ordered pair $(30.0, 60.0)$ since the first vector is 30 degrees counterclockwise from the x-axis and the second vector is 60 degrees counterclockwise from the x-axis. *You should assume that there exists some vector through the origin that does not intersect with any of the objects. This is a very useful assumption for solving this problem with a greedy algorithm.* So, you're given an array A of n ordered pairs representing the angles that define the n objects (each angle is a real number greater than or equal to 0 and less than 360) *and* an angle α representing a vector that does not intersect with any object. For simplicity, you may assume that no two of the given $2n$ angles are the same.

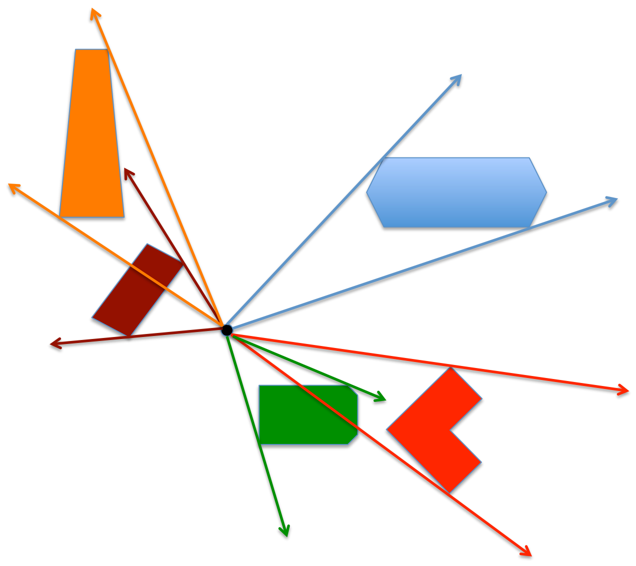

Figure 1: Your space ship (indicated by a black dot) and five scary objects. Each scary object is represented by a pair of vectors representing its angular extent with respect to the space ship.

a. Prof. I. Lai of the Pasadena Institute of Technology claims that the following greedy algorithm minimizes the number of zaps: Find a ray that intersects the largest number of scary objects (breaking ties arbitrarily). Use that ray, remove the objects that it intersects, and repeat until all objects are removed. Describe a small counterexample (you can describe it with angles representing the objects rather than drawing it) for which this greedy algorithm fails. (If you wish, you can exploit the fact that the proposed algorithm breaks ties arbitrarily. Thus, your counterexample might break ties in ways that lead to a suboptimal solution.)

b. Describe a greedy algorithm that determines a smallest set of rays that need to be fired to annihilate all of the given objects. (In the example shown in Figure 1, the solution would comprise three rays.)

c. Prove the correctness of your algorithm using mathematical induction and the safe choice method.

Solution:

a. Counterexamples abound!

b. First, we scan through the array A and add 360 to any angle that is less than α. In this way, we represent angles so that their values are all greater than α. Now, we sort the $2n$ angles from smallest to largest. For each angle, we annotate it with the name of the object (e.g., the index in array A) with which it is associated and indicate whether it is the start angle (θ) or the end angle (ϕ) for that object. Now, we begin with the first entry in the sorted array. We move from left to right in the array until we find the first element representing an end angle. We fire a ray at that angle (adding it to the initially empty set of rays comprising our solution), remove all of the objects that intersect with it, and repeat the process from the next remaining element of the array.

c. The proof is by strong induction on the number of objects, n. The base case is $n = 0$ which is trivial. For the induction hypothesis, assume that the algorithm is correct when there are between 0 and n objects. For the induction step, consider an arbitrary set of $n + 1$ objects and an angle α which doesn't intersect any object. Consider the end angle initially found by our greedy algorithm. Call this angle γ (for "greedy"). We claim that there exists an optimal solution that fires a ray at angle γ. Let OPT denote an optimal solution. If OPT contains a ray at angle γ then this claim is true. If not, let β be the ray with the smallest angle in OPT. Note that $\alpha \leq \beta < \gamma$ since an angle at least as large as α is required to zap the object corresponding to the first element in the sorted array and if $\beta > \gamma$ then the solution fails to zap the object corresponding to the end angle γ. Note that ray γ zaps all of the objects zapped by ray β. Therefore, ray β can be replaced by ray γ. So, we have now established that there exists an optimal solution that includes ray γ. Once we remove all of the objects zapped by γ, the remaining number of objects is between 0 and n and thus, by the induction hypothesis, the greedy algorithm finds the optimal number of rays for the remaining problem. Thus, the greedy algorithm is optimal.

Problem 4: Connect-the-Dots! [30 Points]

Connect-the-Dots is a game that works like this: We are given $2n$ dots on a line. The dots are denoted $d_1, \ldots, d_{2n}$ from left-to-right; n of the dots are black and n are white. Our goal is to match each black dot with a single white dot, minimizing the total length of the horizontal line segments used to match the dots. In general, there may be multiple optimal solutions, but we just seek one. For example, consider the 8 dots in Figure 2 where in this particular case, the distance between a dot and the dot to its right is one unit. This

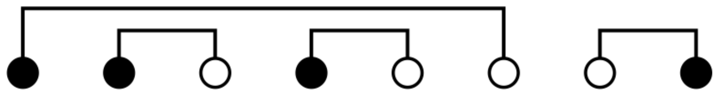

Figure 2: An example of the Connect-the-Dots Game with 8 dots.

figure shows one possible solution of total cost 8 since since the total length of the horizontal segments in this matching is 8. In general, the dots are not necessarily uniquely spaced. Let the distance between dots d_i and d_j be denoted by $\delta(d_i, d_j)$.

a. Describe a simple greedy algorithm for finding an optimal solution to this problem. (Careful: Not every greedy algorithm for this problem will give an optimal solution.)

b. Prove the correctness of your algorithm using mathematical induction and the safe choice method. (*Note:* You may need to do a bit of case analysis in your proof. Our sample solution looks at a few different cases that can arise. You may also need to use the δ distance function as you do the case analyses.)

Solution: The algorithm is just this: Begin at the leftmost dot. Find the first dot to its right that is of the opposite color and match that pair of dots. Remove those two dots and repeat the process. We claim that the algorithm is optimal for any set of n black and n white dots, regardless of the distances between successive dots. Our proof is by induction on n (the number of pairs of dots). When $n = 1$, our algorithm finds the only possible solution and is therefore optimal. For the induction hypothesis, assume that the algorithm is correct for any n pairs of dots and consider an arbitrary set of $n + 1$ pairs of dots. Number the dots $d_1, \ldots, d_{2n}$ from left-to-right. Without loss of generality, assume that the leftmost dot, d_1, is black. The algorithm then finds the nearest white dot, d_i, and matches d_1 to d_i. We first show that this choice is safe in the sense that there must be some optimal solution that matches d_1 to d_i. Assume by way of contradiction that there exists an optimal solution S' that is better than our greedy solution and does not match d_1 to d_i. In that case, d_1 must be matched to a white dot d_j such that $j > i$. But, d_i must be matched to some black dot in solution S'. Let d_k denote the black dot to which d_i is matched. There are three possible cases: $1 < k < i$ or $i < k < j$ or $j < k$.

We now show that in each of these cases, we can modify S' by matching d_1 to d_i and matching d_j to d_k without increasing the total length of the horizontal segments in the matching.

First, consider the case that in S', $1 < k < i$. Then, before modifying S', the matching of d_1 to d_j and d_k to d_i contributes a total of $\delta(d_1, d_k) + \delta(d_k, d_i) + \delta(d_i, d_j) + \delta(d_k, d_i) = 2\delta(d_i, d_k) + \delta(d_1, d_k) + \delta(d_i, d_j)$ and after reassigning d_1 to d_i and d_k to d_j, that pair contributes $\delta(d_1, d_k) + \delta(d_k, d_i) + \delta(d_k, d_i) + \delta(d_i, d_j) = 2\delta(d_i, d_k) + \delta(d_1, d_k) + \delta(d_i, d_j)$ and there is therefore no net change in the total. If $i < k < j$, the matching of d_1 to d_j and d_k to d_i contributes

$2\delta(d_i, d_k)+\delta(d_1, d_i)+\delta(d_k, d_j)$ beforehand and $\delta(d_1, d_i)+\delta(d_k, d_j)$ afterwards and thus reduces the total horizontal lengths. Finally, if $j < k$ then the matching of d_1 to d_j and d_k to d_i contributes a total of $\delta(d_1, d_i) + \delta(d_i, d_j) + \delta(d_i, d_j) + \delta(d_j, d_k)$. But, after matching d_1 to d_i and d_j to d_k, that pairing contributes $\delta(d_1, d_i) + \delta(d_j, d_k)$ which is a net decrease in the total horizontal lengths.

So, we have shown that there exists an optimal solution in which d_1 is matched with d_i. Now, upon removing that pair of dots, we have a problem with n black dots and n white dots. By the induction hypothesis, the algorithm finds an optimal solution to that subproblem and thus our algorithm finds an optimal solution to the given problem instance with $n + 1$ pairs of dots.

Problem 5: (Optional Bonus Problem) The Arithmetic and Geometric Mean [25 Points]

Let $x_1, \ldots, x_n$ be positive real numbers. The *arithmetic mean* of these numbers is defined to be $\frac{x_1+x_2+\ldots+x_n}{n}$ and the *geometric mean* is defined to be $(x_1x_2 \cdots x_n)^{1/n}$. In this problem we show that the arithmetic mean of n numbers is at least as large as the geometric mean of those numbers.

a. Use induction to show that if $x_1x_2 \cdots x_n = 1$ then $x_1 + x_2 + \ldots + x_n \geq n$. (Beware of the induction pitfall mentioned in the handout on writing inductive proofs.)

b. Use this fact to show that the arithmetic mean is at least as large as the geometric mean. (No induction required here; just a little algebra.)